Quaternary-Binary Message-Passing Decoder for Quantum LDPC Codes

Dimitris Chytas, Nithin Raveendran, Asit Kumar Pradhan, and Bane Vasić
Department of Electrical and Computer Engineering,
The University of Arizona, Tucson, AZ 85721, USA.
Email: dchytas@arizona.edu, nithin@arizona.edu, asitpradhan@arizona.edu, vasic@ece.arizona.edu

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10436874

***Abstract*—We introduce a low-complexity message-passing quantum error correction algorithm for decoding Quantum Low-Density Parity-Check (QLDPC) stabilizer codes. The proposed decoder operates on the quaternary stabilizer graph but only exchanges binary messages. This leads to a significantly reduced complexity compared to other quaternary belief propagation (BP) algorithms that pass floating-point messages. The efficacy of the proposed decoder is evaluated by providing decoding examples, performance metrics using Monte-Carlo simulations, and complexity analysis. Despite its reduced complexity, the performance loss of the proposed decoder is modest compared to floating-point parallel quaternary decoders for a Calderbank-Shor-Steane (CSS) code family. In particular, experiments obtained over the $[[1054, 140, 20]]$ lifted product (LP) Tanner code demonstrated that for low error rates (< 0.01), the proposed quaternary-binary message-passing decoder approaches the performance of quaternary BP by converging in almost the same number of iterations while requiring less complex operations. Additionally, for non-CSS codes, our decoder performs similarly as quaternary floating-point decoders despite its lower complexity.**

I. Introduction

Low-Density Parity-Check (LDPC) codes have been shown to have near Shannon-capacity performance and have a wide range of applications, including 5G New Radio (NR) standards, satellite communication, data storage, and image/video compression [1]–[4]. Recently, Quantum LDPC (QLDPC) codes [5]–[8] have become a subject of significant interest in fault-tolerant quantum computation as stabilizer codes beyond traditional topological codes. Recent breakthrough results showed the existence of 'good' QLDPC codes belonging to the Calderbank-Shor-Steane (CSS) family that achieve optimal scaling in terms of both the number of logical qubits and the minimum distance, relative to the code length [9]–[11].

However, having codes with good minimum distance is not enough to realize scalable fault-tolerant quantum computers. We need decoders that have high throughput and achieve a low logical error rate. The high throughput criterion is essential because the qubits have low coherence time. As an illustration, in recent experiments by Google's Quantum Artificial Intelligence team, error correction is performed once in every 921 nanoseconds [12]. In another experiment, researchers perform error correction in every 1.1 microseconds [13]. The need for fast decoders comes from the fact that if the syndrome bits are processed slower than the speed at which they are generated (by measuring the stabilizers), then the running time of quantum algorithms increases exponentially with the T-depth of the circuit [14]. As the number of qubits increases, the high throughput requirement poses serious engineering challenges in designing decoders, a major block in fault-tolerant quantum computers.

Most recent research focuses on designing decoders with a low logical error rate without considering the high throughput criteria [15]–[19]. The Ordered Statistics Decoder (OSD) [15], [20] and Stabilizer Inactivation Decoder [16] involve a post-processing step whose complexity scales polynomially with the number of qubits, hence unsuitable to meet the high throughput criteria [21]. Similarly, the Generalized Belief Propagation (GBP) [17], [18] algorithm achieves low error rate at the expense of higher complexity. Recently proposed Refined Belief Propagation (RBP) decoder [19] decodes over the stabilizer graph (a quaternary graph with edges labeled as X, Y, or Z as given in Fig. 1) and has the same performance as the conventional quaternary BP (BP_4). RBP passes scalar messages instead of quaternary vectors as in BP_4, hence has less complexity. The performance of RBP can be improved further by optimizing the normalization factor through heuristics or greedy approaches [22]. Despite its lower complexity, RBP will not be able to meet the high throughput criteria as the number of qubits scales because it relies on full precision implementation.

To meet the high throughput requirement, the authors in [23] and [24] proposed a sliding window decoder and a parallel window decoder, respectively. In [25], it is shown that iterative hard-decision decoders provide throughput in the order of Gbps while decoding classical LDPC codes without significantly compromising the error rate. In [26], the authors proposed a hard-decision decoder that can provably correct a linear fraction of errors in a linear time. However, the proof relies on the expansion property of the code's Tanner graph, which is not practical neither from the standpoint of code construction nor from the constraints on spatial placement of physical qubits. Motivated by the high throughput of hard-decision decoders, it is interesting to see the possibility of designing hard-decision iterative decoders to decode both CSS and non-CSS quantum codes, which performs similarly to the full-precision decoders. In this paper, we present a low-complexity hard decision iterative decoder, referred to as Quaternary-Binary Message-Passing decoder (QB-MPD). Decoding is performed by exchanging binary messages over the quaternary stabilizer graph. Message passing update rules are derived with the similar spirit of RBP in Section III, along with decoding examples using the 5-qubit

code. We evaluate the complexity of our approach, in terms of both the number of operations and average iterations, by comparing it to RBP, which is currently the least complex full-precision quaternary quantum decoder. The lower complexity of the proposed QB-MPD decoder makes it a suitable solution for hardware-friendly decoding. For instance, at the parity check nodes, we require only the XOR logic instead of complex check operations (see Section V). However, as anticipated with a hard-decision approach, there is a slight decrease in performance as shown in the performance analysis in Section V.

II. Preliminaries

Let $S = \{S_1, \cdots, S_m\}$ be a set of stabilizer generators for a $[[n, k, d_{min}]]$ quantum code, where each generator S_i, $1 \leq i \leq m$ is a n-qubit Pauli operator [27], $k = n - m$ and d_{min} is the minimum distance of the code. The Pauli operators are: $I = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right], X = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right], Y = \left[\begin{smallmatrix} 0 & -i \\ i & 0 \end{smallmatrix}\right], Z = \left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$. The set of generators $\mathcal{S}$ can be represented using a matrix $\mathbf{H}$, called *stabilizer matrix*, whose $(i, j)^{\text{th}}$ element is given by the Pauli operator corresponding to the j^{th} qubit in stabilizer $S_i \in \mathcal{S}$. The commutator between two operators $A, B \in \{I, X, Z, Y\}$ is defined as $[A, B] = AB - BA$. From the definition, it follows that $[A, B] = 0$ when A and B commute, otherwise $[A, B] = 1$. Note that X, Y, and Z operators anti-commute with each other, while I commutes with all the Pauli operators.

We consider a depolarizing channel characterized by channel parameter ϵ (depolarizing error rate), which induces error pattern $\boldsymbol{E} \in \{I, X, Z, Y\}^n$ on the n qubits. In the depolarizing channel, every qubit goes through either a bit-flip (X), phase-flip (Z), or both (Y), each with probability $\epsilon/3$. The probability of no error (I) on the same qubit equals to $1 - \epsilon/3$. The weight of $\boldsymbol{E}$ is defined as the number of non-identity operators in $\boldsymbol{E}$. Unlike classical error correction, the decoder cannot estimate the codeword without perturbing the codeword state. This issue can be circumvented by measuring the syndromes.

Let s_i be an indicator function which indicates the commutativity between S_i and $\boldsymbol{E}$, i.e., $s_i = [S_i, \boldsymbol{E}]$. The vector $\boldsymbol{s} = \{s_1, s_2, \ldots, s_m\}$ is referred to as the syndrome vector. In quantum error correction, the decoder estimates the most likely error pattern $\widehat{\boldsymbol{E}} \in \{I, X, Z, Y\}^n$ given the syndrome vector $\boldsymbol{s} \in \{0, 1\}^m$, and the underlying Tanner graph corresponding to the stabilizer matrix $\mathbf{H}$. The Tanner graph is a bipartite graph with a set of variable nodes and check nodes connected by edges that correspond to non-identity entries of $\mathbf{H}$. An example of a parity-check matrix and its Tanner graph is given in Fig. 1, in which edges with X and Z labels are shown in red and blue colors, respectively (edges with Y labels with green). The decoding process succeeds when the estimated error pattern $\widehat{\boldsymbol{E}}$ is equivalent to the actual error pattern up to some stabilizer (degeneracy effect); otherwise a logical error occurs.

III. Quaternary-Binary message-passing decoder

The first part of this Section introduces symbols and notations used in our decoder, while the next part describes the message-updating rules through a decoding example.

$$\mathbf{H} = \begin{bmatrix} X & Z & Z & Z & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & Z \end{bmatrix}$$

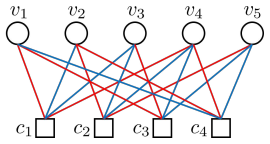

Fig. 1. Stabilizer matrix and Tanner graph of the 5-qubit code. Red and blue edges correspond to X and Z Pauli operators, respectively.

A. Notations

For the rest of the paper, vectors are denoted by bold small-case letters. We denote the natural numbers from 1 to n by [n]. The cardinality of a set $\mathcal{A}$ is $|\mathcal{A}|$. We denote the binary message passed from check node (CN) c_i to variable node (VN) v_j in the ℓ^{th} iteration by $\mu^{(\ell)}_{c_i \to v_j}$. Similarly, the message from VN v_j to CN c_i is denoted by $\nu^{(\ell)}_{v_j \to c_i}$. The degree of variable node v_j and check node c_i is denoted by d_{v_j} and d_{c_i}, respectively. We denote the neighbors of variable node v_j and check node c_i by $\mathcal{M}(v_j)$ and $\mathcal{N}(c_i)$, respectively.

Algorithm 1 Quaternary-Binary Message-Passing Decoder

Input: $\boldsymbol{s}, \mathbf{H}$
Output: $\hat{\boldsymbol{E}}$
Parameters: L: maximum number of iterations, d_v: maximum variable degree.

1) **Initialization:**
   $\ell = 1$, success $= 0$
   Estimated error vector: $\hat{E}^{(0)} = II \ldots I$.
   variable-to-check messages:
   $\nu^{(0)}_{v_j \to c_i} = 0, \boldsymbol{q}^{(0)}_{v_j \to c_i} = (d_v, 0, 0, 0), \boldsymbol{q}^{(0)}_{v_j} = (d_v, 0, 0, 0)$.
2) While $(\ell \leq L)$ and (success $==0$)
   a) **Check node update (CNU)**:
   $$\mu^{(\ell)}_{c_i \to v_j} = s_i \oplus_{v'_j \in \mathcal{N}(c_i)/v_j} \nu^{(\ell-1)}_{v'_j \to c_i} \tag{1}$$
   b) **Variable node update (VNU)**:
   Quaternary messages update:
   $$q^{(\ell)}_{v_j \to c_i}(W) = q^{(\ell-1)}_{v_j \to c_i}(W) + \sum_{c'_i \in \mathcal{M}(v_j)/c_i} \mathbb{1}_{[\mathbf{H}(i',j), W] = \mu^{(\ell)}_{c'_i \to v_j}}, \tag{2}$$
   where $W \in \{I, X, Z, Y\}$.
   Binary variable-to-check message:
   $$\nu^{(\ell)}_{v_j \to c_i} = \begin{cases} 0, & \text{if } \sum\limits_{[\mathbf{H}(i,j), W]=0} q^{(\ell)}_{v_j \to c_i}(W) \geq \sum\limits_{[\mathbf{H}(i,j), W]=1} q^{(\ell)}_{v_j \to c_i}(W) \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$
   c) **Decision vector update**:
   $$q^{(\ell)}_j(W) = q^{(\ell-1)}_j(W) + \sum_{c_i \in \mathcal{M}(v_j)} \mathbb{1}_{[\mathbf{H}(i,j), W] = \mu^{(\ell)}_{c_i \to v_j}}. \tag{4}$$
   d) **Estimated error vector**:
   $$\hat{E}^{(\ell)}_j = \underset{W \in \{I,X,Z,Y\}}{\arg\max}\ q^{(\ell)}_j(W). \tag{5}$$
   e) **Syndrome calculation:**
   if $\sum\limits_{j=1}^{n} [\hat{E}_j, \mathbf{H}(i,j)] \mod 2 = s_i, \forall i \implies$ success $= 1$,
   else $\ell = \ell + 1$.

B. Message-passing rules

The binary message $\mu^{(\ell)}_{c_i\to v_j}$ is assigned to either 0 or 1 such that the stabilizer at CN c_i matches syndrome s_i, given all the extrinsic incoming messages at CN c_i. The binary message $\nu^{(\ell)}_{v_j\to c_i}$ is obtained from $\boldsymbol{q}^{(\ell)}_{v_j\to c_i}$ using Eq. (3). The quaternary vector $\boldsymbol{q}^{(\ell)}_{v_j\to c_i}$ is $\left(q^{(\ell)}_{v_j\to c_i}(I), q^{(\ell)}_{v_j\to c_i}(X), q^{(\ell)}_{v_j\to c_i}(Z), q^{(\ell)}_{v_j\to c_i}(Y)\right)$, whose elements represent a statistics indicating the chances that the error $\boldsymbol{E}^{(\ell)}_j$ commutes with a distinct Pauli operator, given the extrinsic incoming messages to VN v_j and its value $\boldsymbol{q}^{(\ell-1)}_{v_j\to c_i}$ in the previous iteration. Note that vector $\boldsymbol{q}^{(\ell)}_{v_j\to c_i}$ depends on both past and current messages, implying the dependence of $\nu^{(\ell)}_{v_j\to c_i}$ on both past and current messages. This memory attribute leads to improvement in logical error rate as shown in Section V. Next, we define a length four vector $\boldsymbol{q}^{(\ell)}_j = \left(q^{(\ell)}_j(I), q^{(\ell)}_j(X), q^{(\ell)}_j(Z), q^{(\ell)}_j(Y)\right)$, called the decision vector, in which $q^{(\ell)}_j(W)$, for $W \in \{I, X, Y, Z\}$, indicates the chances of error symbol $\boldsymbol{E}^{(\ell)}_j$ being W. From Eq. (4), observe that $\boldsymbol{q}^{(\ell)}_j$ also depends on the current messages and its past value. The decoder message passing rules are summarized in Algorithm 1.

Next, we motivate the message-passing rules of the proposed QB-MPD through an example. Consider the 5-qubit non-CSS code whose parity-check matrix and Tanner graph are given in Fig. 1. This is the smallest stabilizer code that protects against any arbitrary single error on any qubit [28].

Consider the error pattern $\boldsymbol{E} = XIIII$. Observe that all the stabilizer generators except the stabilizer at CN c_4 commutes. Therefore, the syndrome vector is given by $\boldsymbol{s} = (0, 0, 0, 1)$. The decoding process starts by initializing $\widehat{\boldsymbol{E}}^{(0)}_j = I\ \forall j \in [n]$, $\boldsymbol{q}^{(0)}_{v_j\to c_i} = \boldsymbol{q}^{(0)}_j = (4, 0, 0, 0)$, and $\mu^{(0)}_{c_i\to v_j} = 0$ for $\mathbf{H}(i, j) \neq I, i \in [4]$ and $j \in [5]$. Since $\widehat{\boldsymbol{E}}^{(0)}_j = I$ and I commutes with all the Pauli operators, $\boldsymbol{q}^{(0)}_{v_j\to c_i}$ for all the edges is given by $\boldsymbol{q}^{(0)}_{v_j\to c_i} = (4, 0, 0, 0)$, where 4 is the maximum VN degree. By substituting $\boldsymbol{q}^{(0)}_{v_j\to c_i}$ in Eq. (3), $\nu^{(0)}_{v_j\to c_i} = 0$ is obtained for all the edges. Next messages from CNs to VNs are computed using Eq. (1). Since $s_i = 0$ and $\nu^{(0)}_{v_j\to c_i} = 0$ for $i \in \{1, 2, 3\}$, all the outgoing messages from CN 1, 2, and 3 are zero. Since $s_4 = 1$ and the incoming messages at c_4 are zero, the outgoing messages from CN c_4 are one. That is,

$$\mu^{(1)}_{c_4\to v_1} = \mu^{(1)}_{c_4\to v_2} = \mu^{(1)}_{c_4\to v_4} = \mu^{(1)}_{c_4\to v_5} = 1,$$

and all other check-to-variable messages are zero in iteration 1. Next, consider the computation of $\boldsymbol{q}^{(1)}_{v_1\to c_1}$. To better understand the rule in Eq. (2), let us expand the rule for $\boldsymbol{q}^{(1)}_{v_1\to c_1}$. Note that $\mathcal{M}(v_1) = \{c_1, c_3, c_4\}$. Expanding Eq. (2), we get

$$\boldsymbol{q}^{(1)}_{v_1\to c_1}(W) = \boldsymbol{q}^{(0)}_{v_1\to c_1}(W) + \mathbb{1}_{[\mathbf{H}(3,1),W]=\mu^{(1)}_{c_3\to v_1}} + \mathbb{1}_{[\mathbf{H}(4,1),W]=\mu^{(1)}_{c_4\to v_1}}. \quad (6)$$

Recall that $\boldsymbol{q}^{(1)}_{v_1\to c_1}(W)$ represents a statistics that indicate the chances of $\boldsymbol{E}^{(1)}_1 = W$. Also, recall that $\mu^{(1)}_{c_i\to v_1}$ represents whether $\boldsymbol{E}^{(1)}_1$ commutes with $\mathbf{H}(i, 1)$. From Eq. (6), observe that $\mu^{(1)}_{c_i\to v_1}$ for $c_i \in \mathcal{M}(v_1) \setminus c_1$ contributes to $\boldsymbol{q}^{(1)}_{v_1\to c_1}(W)$ only when W satisfy the condition imposed by $\mu^{(1)}_{c_i\to v_1}$. Therefore, the update rule in Eq. (2) is intuitive. By substituting, $\mathbf{H}(3, 1) = X, \mathbf{H}(4, 1) = Z, \mu^{(1)}_{c_3\to v_1} = 0$ and $\mu^{(1)}_{c_4\to v_1} = 1$, Eq. (2) is simplified to the following

$$\boldsymbol{q}^{(1)}_{v_1\to c_1}(W) = \boldsymbol{q}^{(0)}_{v_1\to c_1}(W) + \mathbb{1}_{[X,W]=0} + \mathbb{1}_{[Z,W]=1}. \quad (7)$$

Now by replacing W with the Pauli operators in Eq. (7), we obtain $\boldsymbol{q}^{(1)}_{v_1\to c_1} = (5, 2, 1, 0)$. Next, we motivate the update rule in Eq. (3) through computation of $\nu^{(1)}_{v_1\to c_1}$, which indicates whether $\boldsymbol{E}_1$ commutes with $\mathbf{H}(1, 1) = X$, given $\boldsymbol{q}^{(1)}_{v_1\to c_1}$. We expand the condition in Eq. (3) while computing $\nu^{(1)}_{v_1\to c_1}$, so

$$q^{(1)}_{v_1\to c_1}(I) + q^{(1)}_{v_1\to c_1}(X) \geq q^{(1)}_{v_1\to c_1}(Y) + q^{(1)}_{v_1\to c_1}(Z). \quad (8)$$

Observe that the LHS (RHS) in Eq. (8) clubs the statistics corresponding to the Pauli operators that commute (anti-commute) with $\mathbf{H}(1, 1) = X$. Intuitively, $\nu^{(1)}_{v_1\to c_1} = 0$ since LHS is greater than RHS. Similarly, $\boldsymbol{q}^{(1)}_{v_j\to c_i}$ and $\nu^{(1)}_{v_j\to c_i}$ corresponding to other edges can be computed. Decision vector update at a VN follows a similar argument and can be done using Eq. (4). By following the message-updating steps, it can be shown that the decoder converges in four iterations. It can be also shown that the decoder succeeds on all weight error patterns except $\boldsymbol{E} = IIIYI$.

Algorithm 1 describes the parallel scheduling version of the proposed decoder, where each iteration comprises of first updating all VNs together, then updating the CNs together. The QB-MPD can also be adapted to sequential scheduling wherein either column (or row)-wise updates are performed. In a column-wise scheduled decoder, only a subset of VNs and their adjacent CNs are updated in each iteration. Though sequential scheduling is shown to improve decoding convergence [29], they require a higher latency compared to parallel approaches.

IV. Analysis of QB-MPD for CSS codes

This section investigates the error patterns on which the proposed QB-MPD decoder fails. We start by stating an observation about the QB-MPD.

Remark: *For QB-MPD, the variable-to-check update message sent over an edge with a specific label is only a function of the incoming check-to-variable messages that are sent over edges with the same label.* We skip the proof of the above remark due to the space constraint.

Therefore, while computing the variable-to-check message sent along an edge with a particular label, we only need to consider the incoming check-to-variable messages along the edges with the same edge label while neglecting the rest. For codes like CSS codes, in which edges incident on a CN have the same label, the proposed QB-MPD on the quaternary graph and two QB-MPD decoders that run separately on the X and Z sub-graphs have the same performance.

A. Trapping Sets for QB-MPD

At this point we will introduce the notion of trapping sets and derive sufficient conditions for the cases when cyclic structures in the Tanner graph lead to trapping sets when quantum codes are decoded using QB-MPD.

Definition 1. *According to [29], a trapping set for a syndrome-based iterative decoder is a non-empty set of variable nodes in a Tanner graph G that are not eventually converged or are neighbors of the check nodes that are not eventually satisfied.*

Trapping sets do not allow the decoder to converge for low-weight error patterns and therefore determine the iterative decoding performance in the error floor region [29]. We need additional notations to state the sufficient conditions for a trapping set. Next, we introduce them. Let $\mathcal{E}_v$ be the set of edges connected to VN v. Let $L(e)$ denote the label on edge e. Define $\mathcal{L}(\mathcal{E}) = \{L(e) : e \in \mathcal{E}\}$.

Lemma 1. *Consider Tanner graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{C}, \mathcal{E})$ corresponding to a quantum code. Let $\mathcal{G}_1 = (\mathcal{V}_1 \cup \mathcal{C}_1, \mathcal{E}_1) \subseteq \mathcal{G}$ be the sub-graph induced by a stabilizer generator. Sub-graph $\mathcal{G}_1$ is a trapping set if for every $v \in \mathcal{V}_1$,*

1) $|\mathcal{L}(\mathcal{E}_v \cap \mathcal{E}_1)| = 1$,
2) $|\mathcal{L}(\mathcal{E}_v \cap (\mathcal{E} \setminus \mathcal{E}_1))| = 1$,
3) $\mathcal{L}(\mathcal{E}_v \cap \mathcal{E}_1) \cap \mathcal{L}(\mathcal{E}_v \cap (\mathcal{E} \setminus \mathcal{E}_1)) = \emptyset$.

Proof. The main idea behind the proof is to show that the check-to-variable messages sent by a CN $c \in (\mathcal{C} \cap \overline{\mathcal{C}_1})$ to a VN $v \in \mathcal{V}$ have no effect on the variable-to-check messages exchanged inside the sub-graph $\mathcal{G}_1$. The details of the proof will be given in the longer version of the paper. □

Lemma 1 provides sufficient conditions for a cycle to be harmful under QB-MPD. Consider the sub-graph $\mathcal{G}_1$ induced by any length-8 cycles of the Z and X sub-graph. If the variable degree of both the X and Z sub-graphs is equal to 2, any two error patterns on $\mathcal{G}_1$ in which the non-identity operators are either X or Z satisfy the condition in Lemma 1 and hence, cannot be corrected. These configurations can be classified as $(4, 0)$ symmetric trapping sets with the critical number equal to two, based on the definition of trapping sets in [29]. Hence, the number of these harmful error patterns is $\binom{4}{2} \times N_8 = 6 \times N_8$, where N_8 denotes the total number of 8-cycles.

For example, for the $[[106, 2, 9]]$ code [30], the number of harmful weight-2 patterns equals 636. Fig. 2 shows two instances of harmful 8-cycles in the code's Z sub-graph. An example of the 8-cycle appearing in the Z sub-graph of the $[[106, 2, 9]]$ code consists of the VNs: $v_1, v_2, v_{55}, v_{100}$. Consider the case of a weight-2 error pattern with X errors on v_1 and v_2 (Fig. 2(a)). In this case, the proposed parallel decoder displays an oscillatory behavior. In particular, the error estimate on v_1, v_2, v_{55}, and v_{100} oscillates between $\{X, X, X, X\}$ and $\{I, I, I, I\}$. Similar behavior is also observed for the RBP decoder. That oscillatory behavior is observed if the weight-2 error corresponds to the diagonal nodes of an 8-cycle; otherwise, the decoder gets stuck at the same estimate, which is $\{\hat{E}_1, \hat{E}_2, \hat{E}_{55}, \hat{E}_{100}\}$=$\{X, X, X, X\}$.

V. Performance Evaluation

A. CSS codes

We compare the performance of the proposed QB-MPD decoder against a floating-point quaternary RBP decoder. CSS codes belonging to the generalized bicycle (GB) codes from [30]–[32] and lifted product (LP) codes are chosen for simulation purposes. Each data point in the figures is obtained by simulating the decoders till we observe 100 logical errors. Also, all decoders run for a maximum of 10 iterations under parallel scheduling unless stated otherwise. Fig. 3 depicts the logical error rate when the $[[106, 2, 9]]$ GB code [30] with minimum distance $d_{min} = 9$, $d_c = d_v = 4$ and girth $g = 8$ is decoded by parallel versions of our decoder and RBP. We observe that the dependence of the variable-to-check messages (and decision vector) on their past values improves performance compared to the case where Eq. (2) and Eq. (4) depend only on the incoming check-to-variable messages. By adding the memory component, the decoder becomes more cautious in updating the estimate value of each variable node and therefore updates smaller sets of qubits than the non-memory case. Essentially, the most 'reliable' nodes, i.e., the nodes that have accumulated a certain number of votes over the previous iterations are updated. The same figure also illustrates a result related to Lemma 1 in Section IV. Because we deal with a CSS code, the X and Z sub-graphs can be decoded independently. Hence, we apply our decoder to each sub-graph separately. The quaternary decoder now operates on binary graphs but it has knowledge of the edge labels. For example, when decoding the X sub-graph, the same

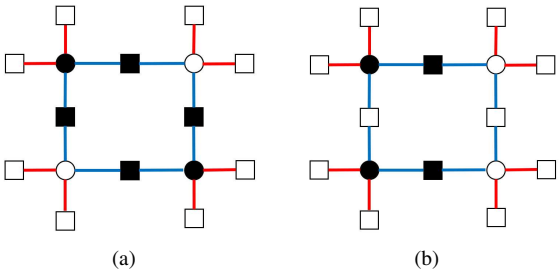

Fig. 2. Two examples of $(4, 0)$ symmetric trapping sets of the $[[106, 2, 9]]$ GB code for QB-MPD. Shaded VNs represent X errors and shaded CNs are anti-commuting stabilizers. Red and blue edges correspond to X and Z Pauli operators, respectively. Both examples satisfy Lemma 1.

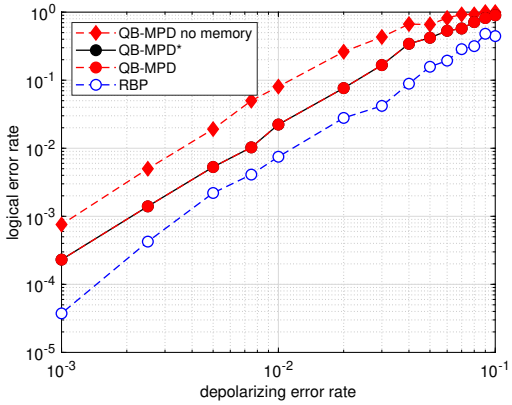

Fig. 3. RBP *vs.* variations of QB-MPD for the $[[106, 2, 9]]$ GB code [30]. *No memory* plot corresponds to omitting the first term of the RHS of Eq. (2) and Eq. (4). QB-MPD* corresponds to applying our decoder to the X and Z sub-graphs of the code separately. QB-MPD* and QB-MPD curves exactly overlap.

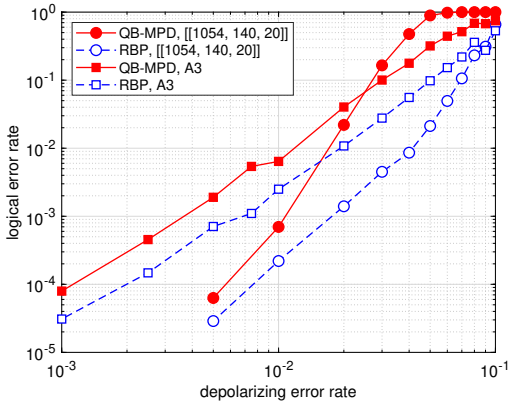

Fig. 4. Logical error rate performance of RBP *vs.* QB-MPD for $[[1054, 140, 20]]$ LP Tanner code and the $[[48, 6, 8]]$ GB code. For low error rates (< 0.01), QB-MPD approaches RBP while requiring less complex operations (Table II).

TABLE I
RBP *vs.* QB-MPD over the number of correctable weight-1 and weight-2 error patterns for various non-CSS codes. Decoders run for 50 iterations.

non-CSS codes	weight-1		weight-2	
	QB-MPD	RBP	QB-MPD	RBP
$[[5, 1, 3]]$	$14/15$	$14/15$	-	-
$[[8, 3, 3]]$	$18/24$	$18/24$	-	-
$[[16, 10, 3]]$	$27/48$	$27/48$	-	-
$[[16, 6, 4]]$	$40/48$	$14/48$	$7/1080$	$32/1080$
$[[11, 1, 5]]$	$33/33$	$32/33$	$5/495$	$17/495$

number of votes is accumulated to both Z and Y operators of the quaternary messages and decision vectors. However, the estimated error vector only contains I and Z operators (I and X operators for the Z sub-graph respectively). QB-MPD* performs exactly the same as QB-MPD, confirming that in the case of CSS codes, decoding on the whole graph using QB-MPD is equivalent to decoding X and Z sub-graphs separately by the same decoder.

Fig. 4 shows the decoding performance of the QB-MPD versus RBP for the $[[1054, 140, 20]]$ LP Tanner code [33] and the $A3$ code ($[[48, 6, 8]]$) [15]. Interestingly, for the case of the LP Tanner code, in the error-floor regime, both decoders display a similar performance.

B. Non-CSS codes

Table I outlines the error correction capability of the QB-MPD and RBP decoder for distance 3, 4, and 5 non-CSS codes (from [6]). QB-MPD, despite its low complexity, has a similar error correction capability as that of RBP for the distance-3 codes and has an even better error correction capability than that of the RBP for the $[[16, 6, 4]]$ code and has almost similar error correction capability as that of the RBP for the $[[11, 1, 5]]$ code.

TABLE II
Complexity in terms of (worst-case) required operations for QB-MPD *vs.* RBP. The number of operations is computed for each outgoing check-to-variable (for CNU) and variable-to-check (for VNU) message. For QB-MPD, operations involve only integers whereas RBP involves only floating-point numbers.

Step	QB-MPD	RBP decoder
CNU	d_c XORs	d_c multiplications
VNU	$2 \times (d_v - 1)$ increments 2 additions 1 subtraction	$2 \times (d_v - 1) + 4$ multiplications 3 additions 2 subtractions

C. Complexity Comparison

We compare the QB-MPD with the RBP approach regarding the number of decoding operations required. Consider the variable-to-check node update, and for clarity, assume all VNs receive $d_v - 1$ messages, half corresponding to X and the other half corresponding to Z labeled edges. Also, assume all CNs receive $d_c - 1$ messages. In the worst-case scenario, QB-MPD increments integers $2 \times (d_v - 1)$ times and makes one comparison per outgoing variable-to-check message. In contrast, RBP needs $2 \times (d_v - 1)$ floating-point multiplications, a normalization, and a subtraction. QB-MPD needs d_c binary XOR computations for each check-to-variable message, while RBP requires d_c multiplications of floating-point messages. The complexity comparison is summarized in Table II. Fig. 5 illustrates the average number of iterations required by each decoder to converge for the case of the $[[106, 2, 9]]$ code and LP Tanner code. Speaking of the $[[106, 2, 9]]$ code, for error rates lower than 0.005, both decoders converge in about 2 iterations. Similar convergence behavior is observed for the LP Tanner code, for which QB-MPD approaches the RBP performance in the error floor region (Fig. 4). Hence, the proposed decoder offers a good performance/complexity trade-off, especially in low error rates where it converges in around the same number of iterations as RBP while requiring far fewer operations and imposing a modest performance loss.

VI. Conclusions and Future Work

Targeting hardware-implementation-friendly message-passing decoding of general QLDPC codes, we proposed a low-complexity, hard-decision decoder operating on quaternary graphs. QB-MPD offers a lower complexity, in terms of the number of operations and the average number of iterations compared to RBP, at the cost of a modest degradation in performance. However, the proposed decoder attains a similar performance to RBP for various non-CSS codes. We also showed that for the specific case of decoding CSS codes with QB-MPD, quaternary decoding is equivalent to decoding the X and Z sub-graphs separately. More detailed investigations are required to evaluate the performance of general non-CSS codes with the proposed decoder. QB-MPD will be used to investigate the error correction capabilities of such codes (as has been done with Gallager-B in the classical case [34]). Another research direction could be the enhancement of the decoding performance of our scheme either by deploying

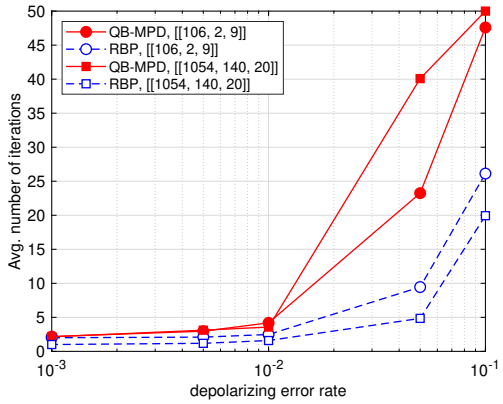

Fig. 5. Complexity of different decoders when used to decode the $[[106, 2, 9]]$ GB code and $[[1054, 140, 20]]$ LP Tanner code. For low error rates (< 0.01), QB-MPD converges in similar number of iterations as RBP while requiring less complex operations (Table II). All decoders run for a maximum of 50 iterations.

post-processing or modifying the existing rules based on the knowledge of quantum trapping sets.

Acknowledgment

This work is supported by the NSF under grants CIF-1855879, CIF-2106189, CCF-2100013 and ECCS/CCSS-2027844, ECCS/CCSS-2052751, and ERC-194158. This work was also funded in part by Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through JPL's Strategic University Research Partnerships (SURP) program. Bane Vasić has disclosed an outside interest in his startup company Codelucida to The University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies. The authors would like to thank Valentin Savin for insightful discussions.

References

[1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
[2] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sept. 1981.
[3] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge University Press, 2008.
[4] H. Li, B. Bai, X. Mu, J. Zhang, and H. Xu, "Algebra-Assisted Construction of Quasi-Cyclic LDPC Codes for 5G New Radio," *Special section on advances in channel coding for 5G and beyond*, vol. 6, pp. 50 229–50 244, Sep 2018.
[5] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.
[6] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997.
[7] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.
[8] N. P. Breuckmann and J. N. Eberhardt, "Quantum low-density parity-check codes," *PRX Quantum*, vol. 2, no. 4, p. 040101, Oct. 2021.
[9] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," *Quantum Inf. and Computation*, vol. 14, no. 15–16, pp. 1338–1372, Nov. 2014.
[10] P. Panteleev and G. Kalachev, "Asymptotically Good Quantum and Locally Testable Classical LDPC Codes," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 375–388.
[11] A. Leverrier and G. Zémor, "Quantum Tanner codes," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022, pp. 872–883.
[12] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, 2023.
[13] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann *et al.*, "Realizing repeated quantum error correction in a distance-three surface code," *Nature*, vol. 605, no. 7911, pp. 669–674, 2022.
[14] D. P. DiVincenzo and P. Aliferis, "Effective fault-tolerant quantum computation with slow measurements," *Physical Review Letters*, vol. 98, no. 2, jan 2007.
[15] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *Quantum*, vol. 5, p. 585, Nov. 2021.
[16] J. D. Crest, M. Mhalla, and V. Savin, "Stabilizer Inactivation for Message-Passing Decoding of Quantum LDPC Codes," in *2022 IEEE Information Theory Workshop (ITW)*, 2022, pp. 488–493.
[17] J. Old and M. Rispler, "Generalized Belief Propagation Algorithms for Decoding of Surface Codes," *Quantum*, vol. 7, p. 1037, Jun. 2023.
[18] N. Raveendran, M. Bahrami, and B. Vasić, "Syndrome-Generalized Belief Propagation Decoding for Quantum Memories," in *IEEE International Conference on Communications*, 2019, pp. 1–6.
[19] K.-Y. Kuo and C.-Y. Lai, "Refined Belief Propagation Decoding of Sparse-Graph Quantum Codes," *IEEE J. Selected Areas in Inf. Theory*, vol. 1, no. 2, pp. 487–498, 2020.
[20] C.-F. Kung, K.-Y. Kuo, and C.-Y. Lai, "On belief propagation decoding of quantum codes with quaternary reliability statistics," 2023.
[21] J. V. Coquillat, F. G. Herrero, N. Raveendran, and B. Vasić, "Syndrome-based min-sum vs OSD-0 decoders: FPGA implementation and analysis for quantum LDPC codes," *IEEE Access*, vol. 9, pp. 138 734–138 743, Oct. 2021.
[22] K.-Y. Kuo and C.-Y. Lai, "Exploiting degeneracy in belief propagation decoding of quantum codes," *npj Quantum Information*, vol. 8, no. 1, Sep. 2022.
[23] P. Das, A. Locharla, and C. Jones, "Lilliput: A lightweight low-latency lookup-table decoder for near-term quantum error correction," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 541–553.
[24] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, "Parallel window decoding enables scalable fault tolerant quantum computation," *arXiv preprint arXiv:2209.08552*, 2023.
[25] B. Unal, A. Akoglu, F. Ghaffari, and B. Vasić, "Hardware implementation and performance analysis of resource efficient probabilistic hard decision LDPC decoders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 3074–3084, 2018.
[26] A. Leverrier, J.-P. Tillich, and G. Zémor, "Quantum expander codes," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, 2015, pp. 810–824.
[27] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. New York, NY, USA: Cambridge University Press, 2011.
[28] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, "Perfect quantum error correcting code," *Phys. Rev. Lett.*, vol. 77, pp. 198–201, Jul. 1996.
[29] N. Raveendran and B. Vasić, "Trapping sets of quantum LDPC codes," *Quantum*, vol. 5, p. 562, Oct. 2021.
[30] R. Wang and L. P. Pryadko, "Collection of codes constructed for "Distance bounds for generalized bicycle codes"." [Online]. Available: https://github.com/QEC-pages/GB-codes
[31] ——, "Distance bounds for generalized bicycle codes," *Symmetry*, vol. 14, no. 7, 2022.
[32] A. A. Kovalev and L. P. Pryadko, "Quantum Kronecker sum-product low-density parity-check codes with finite rate," *Phys. Rev. A*, vol. 88, p. 012311, Jul. 2013.
[33] N. Raveendran, N. Rengaswamy, A. K. Pradhan, and B. Vasić, "Soft Syndrome Decoding of Quantum LDPC Codes for Joint Correction of Data and Syndrome Errors," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2022, pp. 275–281.
[34] S. K. Chilappagari and B. Vasić, "Error-correction capability of column-weight-three LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2055–2061, 2009.